Cost-Effective Telemetry and Command Ground Systems Automation Strategy for the Soil Moisture Active Passive (SMAP) Mission

Joshua S. Choi* and Antonio L. Sanders†
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109

Soil Moisture Active Passive (SMAP) is an Earth-orbiting, remote-sensing NASA mission slated for launch in 2014.‡ The ground data system (GDS) being developed for SMAP is composed of many heterogeneous subsystems, ranging from those that support planning and sequencing to those used for real-time operations, and even further to those that enable science data exchange. A full end-to-end automation of the GDS may result in cost savings during mission operations, but it would require a significant upfront investment to develop such comprehensive automation. As demonstrated by the Jason-1 and Wide-field Infrared Survey Explorer (WISE) missions, a measure of "lights-out" automation for routine, orbital pass ground operations can still reduce mission cost through smaller staffing of operators and limited work hours. The challenge, then, for the SMAP GDS engineering team is to formulate an automated operations strategy—and corresponding system architecture—to minimize operator intervention during operations, while balancing the development cost associated with the scope and complexity of automation. This paper discusses the automated operations approach being developed for the SMAP GDS. The focus is on automating the activities involved in routine passes, which limits the scope to real-time operations. A key subsystem of the SMAP GDS—NASA's AMMOS Mission Data Processing and Control System (AMPCS)—provides a set of capabilities that enable such automation. Also discussed are the lights-out pass automations of the Jason-1 and WISE missions and how they informed the automation strategy for SMAP. The paper aims to provide insights into what is necessary in automating the GDS operations for Earth satellite missions.

I. Introduction

A space mission *ground data system* (GDS) is typically a conglomerate of several different subsystems that interact with each other in ways that achieve mission objectives. Top-level goals for a GDS that serve an unmanned spacecraft include providing human operators on Earth the ability to remotely operate the spacecraft and to ensure delivery of data—whether it is science, intelligence, or other—from the flight system to customers. A GDS can have a complex anatomy of heterogeneous subsystems, and quite often these individual subsystems are designed, developed, and delivered by different engineering teams or even organizations. The individual subsystems perform specific functions involved in mission operations, such as, among others: providing capabilities for mission planning and sequencing; analysis of spacecraft navigation and maneuver planning; instrument data capture; downlink, processing, storage, and distribution of flight system engineering telemetry; spacecraft commands generation, translation, and transmission. The last two sets of capabilities mentioned mainly support a GDS use case commonly referred to as *real-time operations.*

Minimizing the monitoring and intervening required by human operators in real-time operations can result in significant savings in mission operations cost. Two recent Earth-orbiting missions have demonstrated this: Jason-1§

---

* AMPCS Software Engineer, Ground Systems Engineering Section, 4800 Oak Grove Drive, M/S: 301-480.
† SMAP GDS System Engineer, Ground Systems Engineering Section, 4800 Oak Grove Drive, M/S: 321-420.
‡ The SMAP mission has not been formally approved by NASA. The decision to proceed with the mission will not occur until the completion of the National Environmental Policy Act (NEPA) process. Material in this document related to SMAP is for information purposes only.
§ *Jason-1* is a joint mission between NASA and France's Centre National d'Etudes Spatiales. It is an oceanography mission to monitor global ocean circulation. It is in operation since 2001.

and Wide-field Infrared Survey Explorer (WISE)[**], both of which are managed and operated by Jet Propulsion Laboratory (JPL) in Pasadena, California, United States. The reduction in operator man-hours was achieved through *automation* of a set of decisions and actions involved in real-time operations. The level of automation realized is thorough enough that routine engineering operation of the missions can be left unattended—"lights-out"—by a human operator, except when anomaly conditions are encountered and operator action is inevitable.[††]

JPL is the development organization for one of NASA's next Earth satellite mission, Soil Moisture Active Passive (SMAP), which is slated for launch in November 2014.[1] The mission is currently in Phase C—early design and development. JPL will also manage and operate the mission post-launch. The SMAP mission shares in common many characteristics as those of Jason-1 and WISE. One of the mission objectives is to achieve similar cost savings as done by Jason-1 and WISE, through automation of real-time operations. However, the SMAP mission has selected a different GDS subsystem element in place of the one that provides the automation services for Jason-1 and WISE.[‡‡] SMAP GDS will use NASA's AMMOS Mission Data Processing and Control System (AMPCS) to provide telemetry processing, storage, reporting, display, and a subset of automation capabilities for real-time operations.[§§]

Mission costs are incurred not only during its operations phase but also during design and development. Therefore, in formulating an automation strategy for real-time operations, system engineers must investigate how much existing capabilities the mission GDS can inherit from the subsystems used in other previous and current mission GDSs, determine how much robustness and what level of autonomy should be required of the automation, all the while taking into account the development cost that satisfying each system requirement will incur. SMAP mission is committed to establishing an automation strategy that balances the level and complexity of automation with the cost that the development effort for it will incur.

In the following sections, we discuss the low-cost automation strategy being pursued for the SMAP mission, specifically for real-time operations within its GDS. Much of the automation engineering depends on the characteristics of SMAP mission itself. We also give an overview of the mission's entire GDS architecture, and then a more in-depth look into the key GDS subsystems involved in telemetry and command automation. Automation goals for the SMAP mission is strongly influenced by those achieved by Jason-1 and WISE missions, but there are new challenges to face with the SMAP mission. We also consider design scenarios that can push the automation level toward true lights-out.

II. SMAP Mission Characteristics

SMAP is a mission recommended by the National Research Council's Earth Science Decadal Survey Panel to NASA, to directly measure Earth surface soil moisture and freeze-thaw states using L-band radar and radiometer instruments. The spacecraft itself will travel around Earth in a low-Earth, sun-synchronous orbit (LEO, SSO). This orbit will be 680 kilometers in altitude. SMAP is a three-year baseline mission. NASA has tasked JPL with the responsibility of the mission's development. Overall mission cost is budgeted to be approximately $700 million.

Spacecraft communication services will be provided by Goddard Space Flight Center's (GSFC) Near Earth Network (NEN) and Space Network (SN). Per orbit, there will be one, two, or three opportunities for communication with the spacecraft, or *passes*. Each pass is relatively short in duration, ranging only from 5 to 10 minutes. The spacecraft will downlink raw science data at a rate of 130 Mbps. Science data will be transmitted via X band while engineering telemetry and commanding will be transmitted via S band. Once science data is downlinked, there will be an approximately one-hour latency for its delivery to the Science Data System (SDS) at JPL. NASA's Earth Observing System (EOS) Data and Operations System (EDOS) will serve as the repository for all downlinked Level 0 (raw) science data and it will provide them to SMAP GDS.

Resources for real-time operations will be limited. SMAP mission intends to staff a small operations team, in line with that of Jason-1 and WISE missions. Operations facility itself is limited in space, which consists of a room

[**] *WISE*, launched in 2009, is a NASA mission to survey the entire sky at infrared wavelengths to uncover previously unseen cosmic objects. After scanning the entire sky twice, it was put into hibernation mode in February 2011.

[††] In mission context, these operators are formally called Flight Project Mission Controllers, Flight Controllers in short, or simply by the call sign "ACE".

[‡‡] The GDS subsystem that provides automation of engineering operations for Jason-1 and WISE missions is JPL Earth Science Mission Center (ESMC).

[§§] AMPCS is a software system that is part of the Advanced Multi-Mission Operations System (AMMOS) tools catalog. AMMOS itself is managed by the Multimission Ground System and Services Office (MGSS) at JPL.

situated on the first floor of Space Flight Support building at JPL. The operations team will not have a round-the-clock work schedule, and hence SMAP's routine real-time operations must be able to run unattended.

III. Telemetry and Commands During a Pass

To understand what is required of real-time operations automation, we must first examine the actions that a human operator would take during a SMAP spacecraft pass. There is a pass opportunity when the SMAP spacecraft travels on a path that is within the communication range of a ground station antenna. Per orbit, pass opportunities may overlap, which is when more than one ground station are able to communicate with the spacecraft. Despite this possible pass overlap, station schedules will ensure that a maximum of one pass at any given time will be involved in real-time operations. There will be at least one minute of cut-off period between station switchovers.

During pass operations, it is important that downlink real-time engineering telemetry interface with the station is always established before the uplink interface. This ensures that telemetry containing direct acknowledgements to uplink products or indirect signals through state-changes in the telemetry are not missed.

Other missions such as Jason-1 and WISE were able to use station monitor data to determine that downlink and uplink communication links with the flight system have been established. The SMAP mission, because it uses NEN services, will not receive station monitor data but will instead determine that a closed-loop communication with the flight system has been established by examining the command counter in the real-time engineering telemetry. In the beginning of the pass period, the operator, through the GDS, can radiate periodic *No Operation* commands, which do not cause any state changes in the flight system except for the commands-received counter, which will increment. The real-time engineering telemetry that reflects the commands-received counter value in the flight system, as it increments, will serve as evidence that the flight system is able to receive commands from ground, indicating that a closed-loop communication link has been established.

There are over 200 engineering health telemetry (EHA) channels defined so far for the SMAP flight system, in its current stage of development. During real-time operations, the operator will not monitor the entire set of channels, which includes many low-level indicators of the flight system state. Particularly for routine operations, the operator may only be concerned with those channels that indicate that the pass activity itself is proceeding normally, such as the commands-received counter, and those channels that trigger alarms ("red" alarms, in particular), which may be a signal that the flight system has entered or approaching an unsafe state.

Once confirming that the closed-loop communication link has been established, the operator can then initiate the radiation of real uplink products to the flight system. Routine uplink products to radiate during a pass include *ephemeris updates* for the spacecraft and on-board data *retransmission/deletion* commands. The retransmission commands may be targeted for either the *recorded engineering telemetry* (as opposed to real-time) or *science data* still kept in on-board memory. The parameters to be embedded in retransmission commands, that specify which segments of data need to be retransmitted, are determined by analyzing *gap reports*. In contrast with a number of other missions, SMAP spacecraft does not require targeting of its instruments and hence eliminates the need to send instrument operations commands. Other, non-routine uplink products may also be sent during a pass, such as *flight software loads* or *patches* and *instrument look-up tables (LUTs)*. The operator may also command the flight system with routine but infrequent products, such as *background sequence loads* and their *activation commands*, *on-board file management commands*, *orbit trim maneuver (OTM) commands*, *critical process commands* (e.g. safe-mode recovery), and other ad-hoc commands as needed.

IV. Activity Between Passes

Before a pass begins, the NEN (or SN) station will initiate a pair of network socket connections to SMAP Mission Operations Center (MOC) at JPL, one for downlink and one for uplink. Exactly when these connection attempts will occur are deterministic, based on a pre-established station schedule in the form of a pass list.[***] The operator would need to start up the GDS telemetry and command subsystems (or verify that they are operational and ready) before the station attempts to connect, to ensure that communication links are in place according to schedule. The general rule is that the GDS telemetry and command subsystems should be ready for inbound station connection attempts 5 minutes prior to the start of pass. After a pass ends, the station will stop flowing telemetry data and will terminate the connections, also according to the pass list. Depending on the operations procedure, the operator may stop and bring down the GDS telemetry and command subsystems at this time.

[***] A pass list entry will specify the start time of the pass, its end time, the time of the first expected telemetry data receipt, the time of the last expected telemetry data receipt, orbit number, and so forth.

Two types of reports are made available to SMAP GDS after a pass ends: *science telemetry gap report* and *pass report*. Gap reports are important pieces of information because segments of telemetry can be lost or corrupt during downlink, in which case the missing telemetry must be retransmitted from the flight system. Gap reports profile these missing segments in data. These reports are generated by the stations and are available from EDOS 10 minutes after a pass finishes. Once a pass is finished, the next pass should start in about 100 minutes. We are thus allowed to make an operational assumption that the gap report from the previous pass will always be available on time before the next pass begins. The retransmission commands should specify the frame location parameters determined from analyzing the gap reports. The other type of report, pass reports, provide metadata of the pass activity itself. Routine real-time operations include obtaining these reports after completion of a pass, analyzing them, and deciding what actions need to be taken either immediately or during the following pass or passes.

Unlike the real-time engineering telemetry that is received directly from the station during a pass, *recorded engineering telemetry* is received post-pass from EDOS. These recorded telemetry, in the form of files, need to be pulled from the remote repository, processed (including determination of alarm conditions), and stored for later analysis by mission personnel.

V. Ground Data System Architecture

To support all of the in-pass and between-passes activities, SMAP GDS consists of numerous subsystems that provide specific and necessary services. Figure 1 shows a functional view of the SMAP GDS architecture. The GDS, as a whole, interfaces with two external systems, which is the *flight system* onboard the SMAP spacecraft and the *Science Data System (SDS)*, which is a ground system that handles the science portion of the mission.

The GDS architecture integrates all of the subsystems required to completely support mission operations. These subsystems provide various services: mission planning, sequence generation, flight system modeling, station scheduling, schedule distribution, spacecraft analysis, orbit determination, trajectory analysis and planning, maneuver design, instruments analysis, NAIF SPICE kernels[†††] generation and distribution, time correlation, Level 0 data capture and distribution, science data gap reporting, generation of products for science support, common data exchange platform, and more. For directly supporting real-time operations, subsystems are required to: capture, process, display and generate reports on engineering telemetry, translate and radiate flight system commands, generate gap reports for missing recorded engineering data, generate retransmission commands based on the gap reports, and track on-board system parameters. Manual operation of all of these GDS activities is complex and incurs much cost.

Although not discussed in this paper, supporting test activities during pre-operation is also an important requirement on the GDS. The GDS therefore includes subsystems to provide workstation support for flight software development, flight system testbed environments, and Assembly, Test, and Launch Operations (ATLO).

A full, centrally controlled, end-to-end, lights-out automation of the entire GDS with its subsystems would require an immense design and development effort. Such an effort entails high costs for the mission. In designing an automation strategy that would keep development costs relatively low while reducing costs during operations, we must limit the scope of automation to those activities and subsystems that have high return on investment. Accordingly, SMAP mission has focused on the automation of routine real-time operation activities and the functional elements that directly support them: *Engineering Data Processing Element (EDPE)* and the *Command Processing Element (CPE)*. Centrally automating these elements thereby requires a new functional element: *Automated Operations Element (AOE)*. There are routine activities that should be automated in other elements also, such as the aforementioned Level 0 data capture and distribution, time correlation, SPICE kernel generation, and so forth. However, rather than charging the AOE to control all the automation in the GDS, non-real-time operations activities can be self-automated or controlled by separate automation drivers.

[†††] *SPICE* is an information system by NASA's Navigation and Ancillary Information Facility (NAIF), and it assists with planning and interpreting scientific observations from instruments in space. SPICE *kernel files*, or simply *kernels*, contain navigation and other ancillary data.[2]

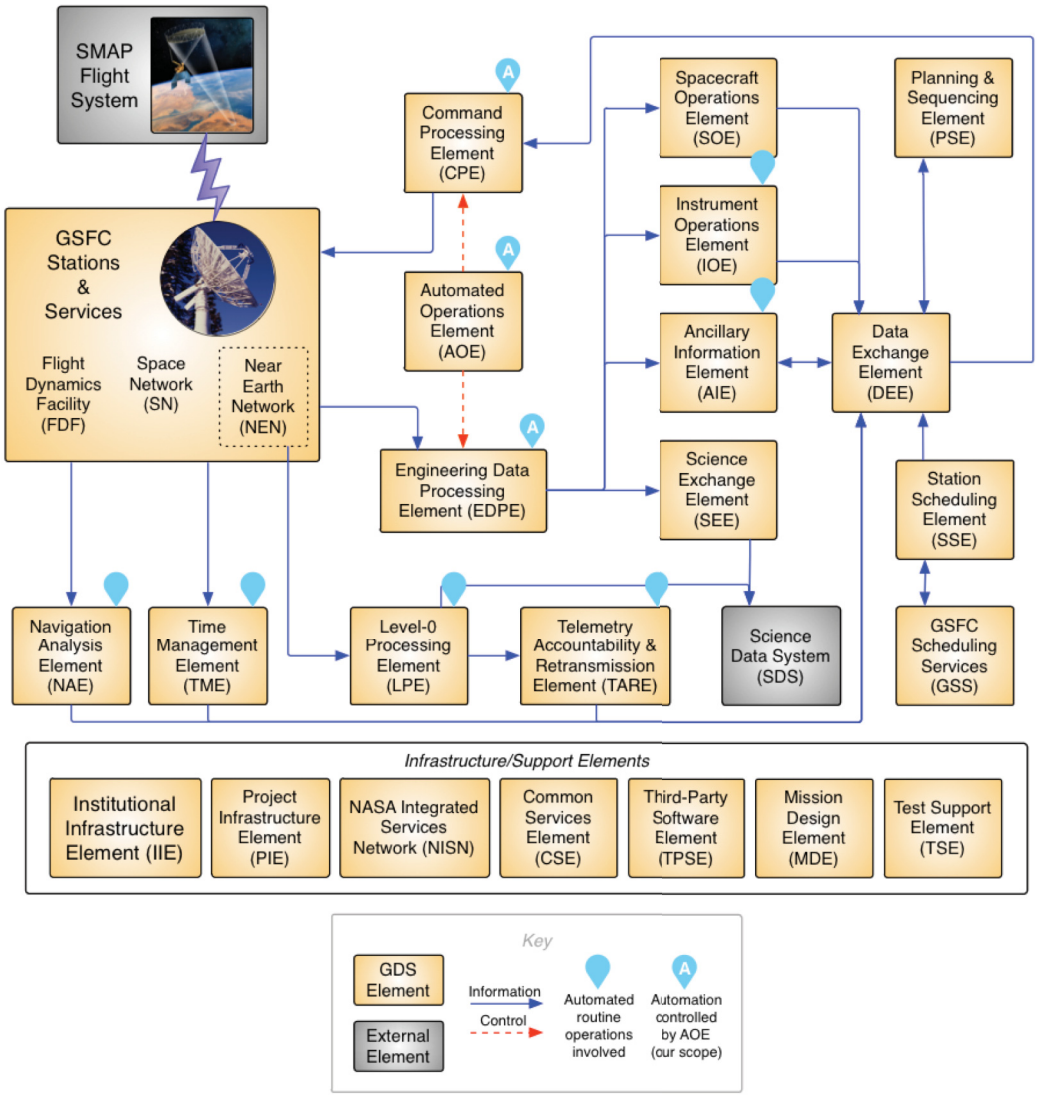

Figure 1. Functional view of the SMAP GDS architecture. *Although many GDS functional elements will automate routine operations activities, the* Automated Operations Element (AOE) *will not directly control all of them. We limit our discussion to the automation of routine* real-time operations *performed by the* Engineering Data Processing Element (EDPE) *and the* Command Processing Element (CPE)*, which are directly controlled by AOE.*

Another factor in controlling mission costs on development effort is the level of *inheritance.* Subsystems incorporated into SMAP GDS generally fall into one of these categories: (1) multimission systems, which have been designed for use by different mission GDSs, hence require little or no new development; (2) legacy systems, which were originally produced for other mission GDSs but may be reused inside the SMAP GDS, perhaps with modifications; and (3) newly developed systems. The EDPE, whose functions are served by AMPCS, has a high level of inheritance due to AMPCS's multimission design and existing capabilities. The CPE functions are served by the Command Preparation & Delivery (CPD) subsystem, which is developed by the Tracking, Telemetry, and Command (TT&C) End-To-End Services office of the Deep Space Network (DSN). Many of CPD's features are being developed to satisfy SMAP mission's requirements, hence there is a moderate level of inheritance for this element. For the AOE, AMPCS already includes a central feature for real-time operations automation called *MPCS Test Automation Kit*, or *MTAK*. Utilizing this existing capability, the AOE benefits from a measure of inheritance.

VI. AMPCS, CPD, and MTAK

AMPCS is a Java and Python-based software suite that provides telemetry processing and storage, alarm processing, display of telemetry through graphical and text user interfaces, data query services and reporting, and automation. It also has basic commanding capabilities that allow users to send immediate commands, Spacecraft Command Message Files (SCMFs), file loads, and raw data files, optionally via Command Link Transmission Units (CLTUs). Figure 2 shows the high-level AMPCS architecture. Each rectangular component represents a separate process, allowing AMPCS to run modularly while exchanging real-time information via the Java Message Service (JMS) or sharing persistent data through a database management system (AMPCS uses MySQL for this purpose).

The basic commanding capabilities provided by AMPCS may be suitable for supporting flight software development as a test tool, but operational commanding has stringent requirements such as security, command verification, radiation ordering, and more. The CPD is the operations-grade uplink subsystem to satisfy those requirements. Although CPD itself is a fully featured command subsystem with its own user interfaces, an interface has been established between the AMPCS and CPD subsystems that will allow users to enter uplink requests to CPD via AMPCS. This interface also allows AMPCS to fully reflect the uplink request statuses and the radiation list in the CPD subsystem via AMPCS user interfaces. As a result, basic routine operations can be done through just the user interfaces that AMPCS provides.

MTAK is an application programming interface (API) framework for the Python programming language that provides a scriptable interface to AMPCS's telemetry and command features. It was originally developed to allow flight system testbed users to automate their tests, which not only saves time and effort, but also indispensable for verification of the flight system through automated regression testing.

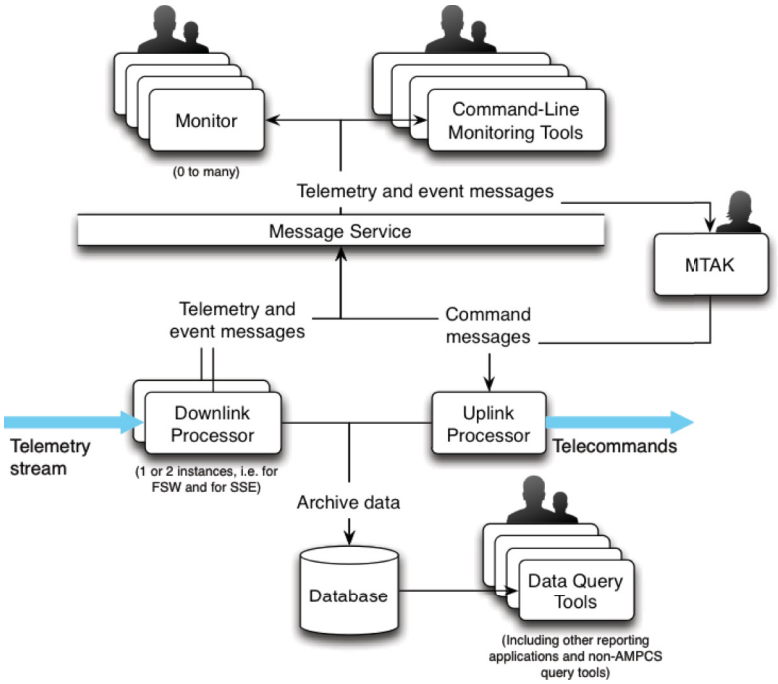

Figure 2. High-level AMPCS architecture. *Each component above run as separate processes, using the Message Service and Database to share information.* MPCS Test Automation Kit (MTAK)*, originally developed to support flight system testers, is key to SMAP GDS's real-time operations automation. (*FSW *and* SSE *are acronyms for* Flight Software *and* Simulation & Support Equipment*, respectively.)*

Using MTAK, user operations such as sending commands (or SCMFs, file loads, raw data files), logging of user-generated messages, and monitoring of telemetry (EHAs, EVRs[‡‡‡], and data products) can be programmed. Scripts using MTAK API require an AMPCS *session* to connect to. An AMPCS session is an actively executing downlink and/or uplink AMPCS run, and all sessions are identified by unique session IDs. Through proxy mechanisms, an MTAK script will attach itself to an AMPCS session, subscribe to telemetry objects that the AMPCS Downlink Processor publishes on the message bus, and initiate transmission of uplink products (e.g. commands) via the AMPCS Uplink Processor. Information exchange between the MTAK script and the active AMPCS Downlink and Uplink Processors are done via the Message Service.

[‡‡‡] *EVR* stands for *Event Record*. It is a type of telemetry that communicates discrete events in the flight system. Reliable processing and monitoring of EVRs is extremely important, as highly critical states (i.e. *FATAL*-level events) in the flight system are communicated to the ground in EVRs.

VII. Automation Strategy

The AOE has a set of subsystem (Level 4) requirements that it needs to satisfy, as shown in Table 1. Many of these requirements are satisfied through combination of pre-existing features in AMPCS and the use of MTAK API. Following are some properties that allow the requirements to be satisfied with minimal new development for SMAP GDS.

A. Generic Python Scripting with MTAK

Table 1. Level 4 subsystem requirements and their rationales for the AOE and its subservices. *(This table lists only the key items and is not an exhaustive list.)*

Subject	Requirement	Rationale
The AOE shall	send navigation ephemeris updates to the spacecraft.	
	allow authorized users to manually edit the order of uplink products.	
	be remotely operable by authorized users.	
	monitor its processes and provide health and status notifications.	
	read tracking pass information from the pass list.	The pass list contains the station pass times which are needed for automation.
	allow the operator to select/deselect scheduled passes for automated processing from the pass list.	Operator needs to have the control to keep a pass automated or perform manually.
	perform pass automation that is configurable via a configuration file.	Pass automation parameters will need to be adjusted during the life of the mission; having this information in configuration files reduces the need to update scripts.
	establish telemetry and command connections to the station interface.	
	display to the operator a visible status indication of the telemetry and command connections to the station interface.	The operator, when present, should see an indication that automation is performing correctly.
	display to the operator a visible, real-time indication of the number of frames received from the station interface.	The operator, when present, should see an indication that telemetry is being received as expected.
	display to the operator a visible, real-time indication of the number of commands sent to the station interface.	The operator, when present, should have an indication that commands are being sent as expected.
	automatically acquire and process recorded engineering data from the station interface after a tracking pass.	
	display to the operator and log to a file the automation status during a tracking pass.	The operator, when present, should have real-time visibility into the automation processing. This information should also be logged.
	allow the operator to disable pass automation at any time.	Pass automation is intended for routine passes. For non-routine activities and anomalous situations, the operator should be able to manually conduct the pass.
	allow the operator to manually abort the automation for an active pass.	Same as above.
	provide a scripting service to support automation of tests and operations.	
The AOE scripting service shall	execute user-defined scripts.	
	execute operating system command-line statements and evaluate returned status.	
	send spacecraft commands in the testbed and ATLO environments.	
	access and evaluate information from telemetry.	
	provide logic and branching constructs.	
	write script execution status information to a log.	
	notify users of script execution progress.	
	send Ground Support Equipment (GSE) commands in the testbed and ATLO environments.	
The AOE timed execution service shall	execute user-defined functions at user-defined times based on relative or absolute time.	
The AOE application service shall	generate periodic reports consisting of: (1) plots of selected parameters and time ranges, (2) comma-separated vales (CSV) files of selected parameters and time ranges, and (3) ASCII text logs of EVR and alarms for selected time ranges.	

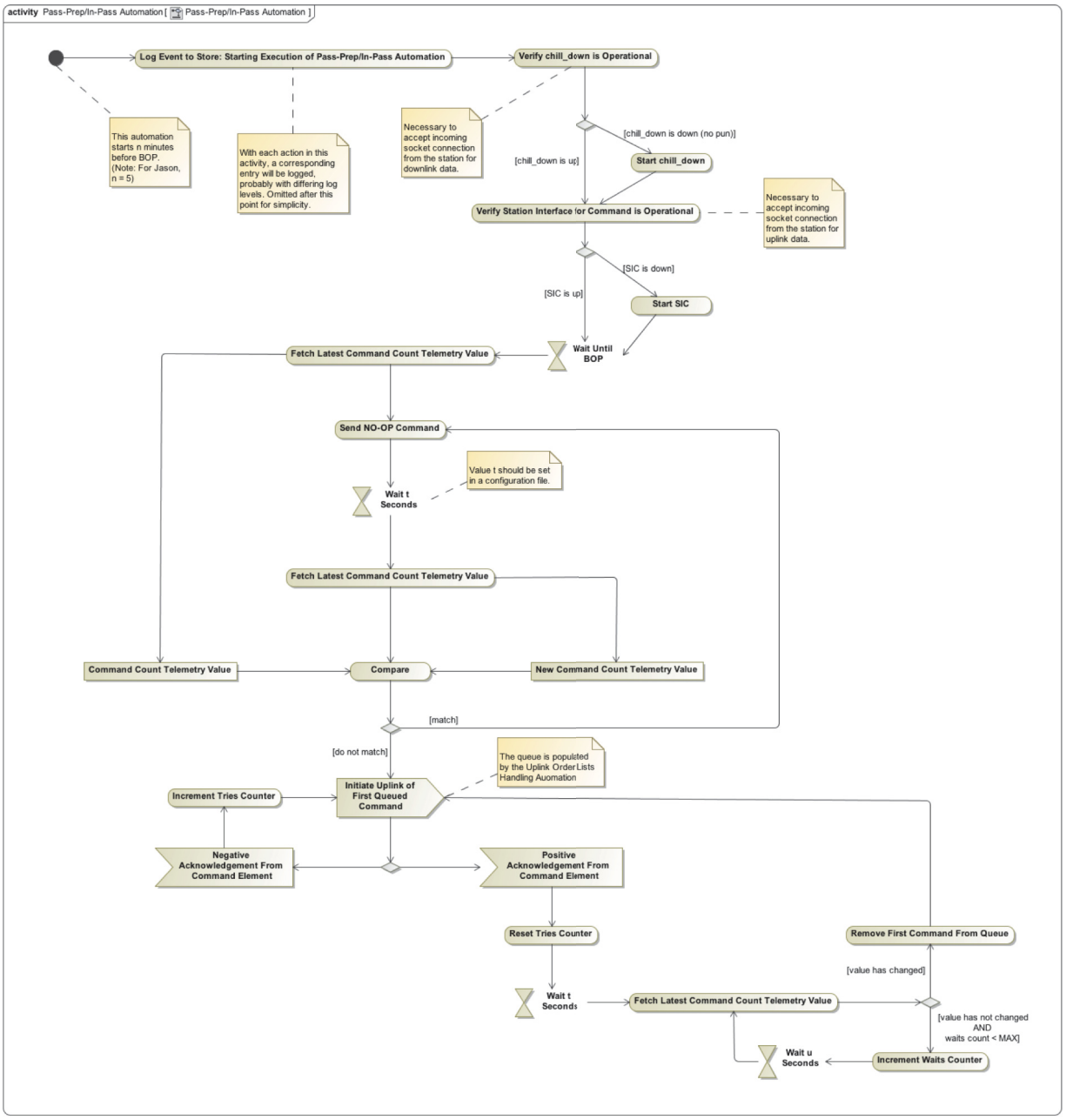

Figure 3. Flowchart of activities to be automated for routine real-time operations. *The automation can be implemented in an MTAK script. (Note: In the diagram,* chill_down *refers to the AMPCS Downlink Processor;* SIC *is acronym for* Station Interface for Command*.)*

MTAK API calls can be embedded in general-purpose Python scripts, and scripts containing MTAK API calls may be invoked as subprocesses by other scripts and executables. This enables starting the execution of the AMPCS Downlink and Uplink Processors, establishing connection to the station (by configuring and initializing the AMPCS Processors to listen for incoming socket connections), sending *No Operation* commands, waiting for telemetry updates, sending ephemeris updates and retransmission commands (assuming they are made available, pre-pass, wholly by the responsible GDS elements), and other tasks to be scripted at operating system level. Python is a high-level, highly readable programming language, and writing and maintaining MTAK-based automation scripts do not incur high development costs. This has been demonstrated by NASA's Mars Science Laboratory (MSL) mission, whose Workstation Test Set (WSTS), Flight Software Internal Testing (FIT), and Testbed activities using MTAK are still ongoing since their first use in 2007.

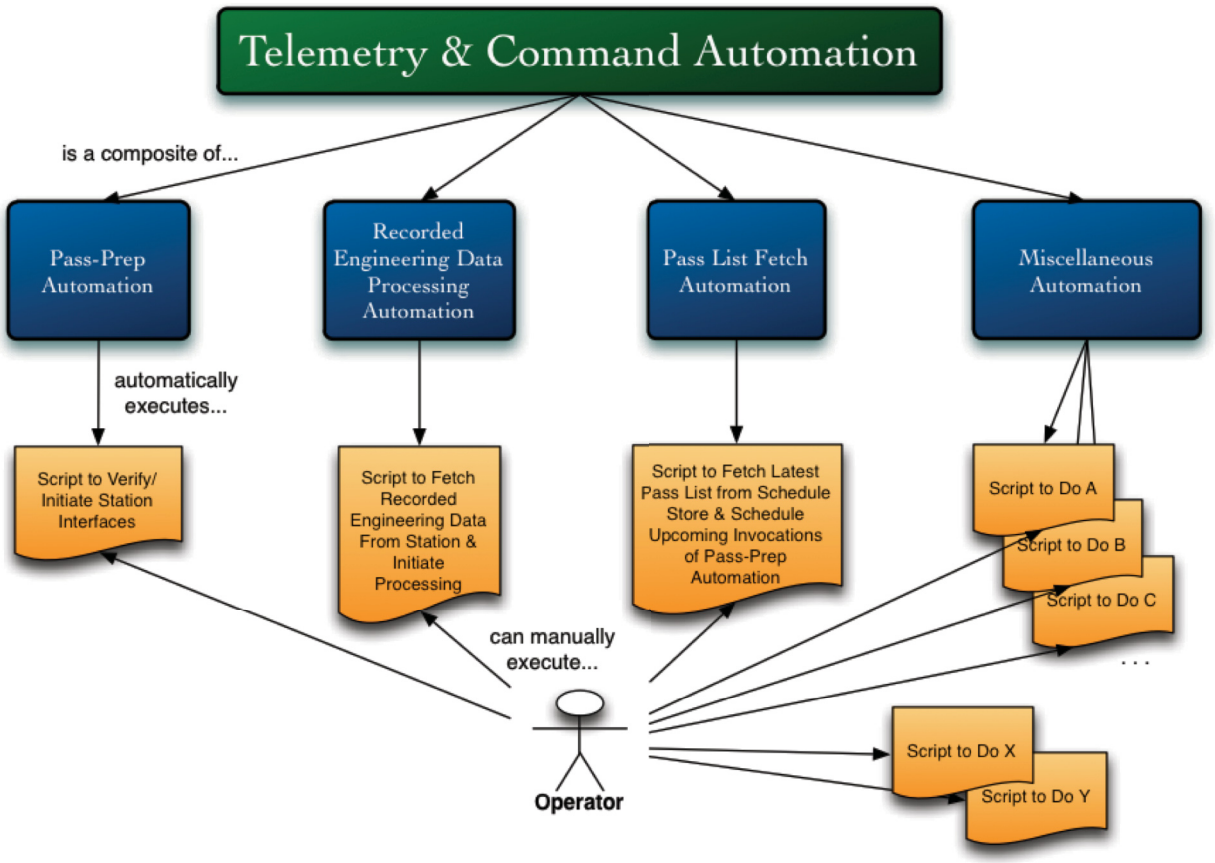

Figure 4. Categorical breakdown of routine real-time operations automation. *AOE's* scripting service *allows automation to be fully modularized and scalable.*

B. AMPCS User Interface in Real-Time Operations

Even with successful automation, the operator should have full access and visibility into its activities and status. The AMPCS user interface satisfies these requirements. Figure 5 and Figure 6 show a pair of examples of AMPCS graphical user interfaces (GUIs). Vital information that operators need to see, such as inflowing frame counts, uplink products sent, EHA data trends, and FATAL EVRs are observable in real-time during a pass.

C. AMPCS Off-Line Processing

AMPCS is designed to work with generic telemetry input (and command output) connections. For receiving and processing real-time telemetry from a station during a pass, the input connection used is a TCP/IP socket. However, AMPCS can also use databases or operating system files as telemetry sources, as part of its *data playback* capability.

Recorded engineering data will not be streamed directly to AMPCS Downlink Processor from the station that received it. It will be replicated to SMAP MOC at JPL via the File Transfer Protocol (FTP). AOE can take advantage of AMPCS's data playback feature, namely by using the file input as the telemetry source, to process the recorded engineering data, generating alarm conditions as AMPCS detects them, and producing reports. The AMPCS instance (or instances, since many can be started and run simultaneously) that handles recorded engineering data can run independently of the instance handling the real-time telemetry from the station. Using generic scripting (not requiring MTAK), fetching and processing of recorded engineering data can be automated as a separate module within AOE.

D. AMPCS Command-Line Features

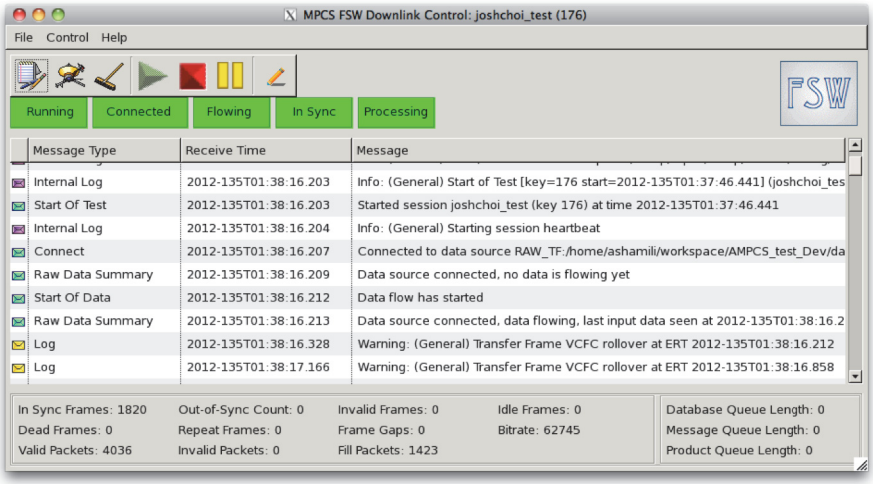

Figure 5. AMPCS Downlink Processor GUI. *Operators can monitor telemetry stream's statistical information, such as the number of frames received, using the GUI.*

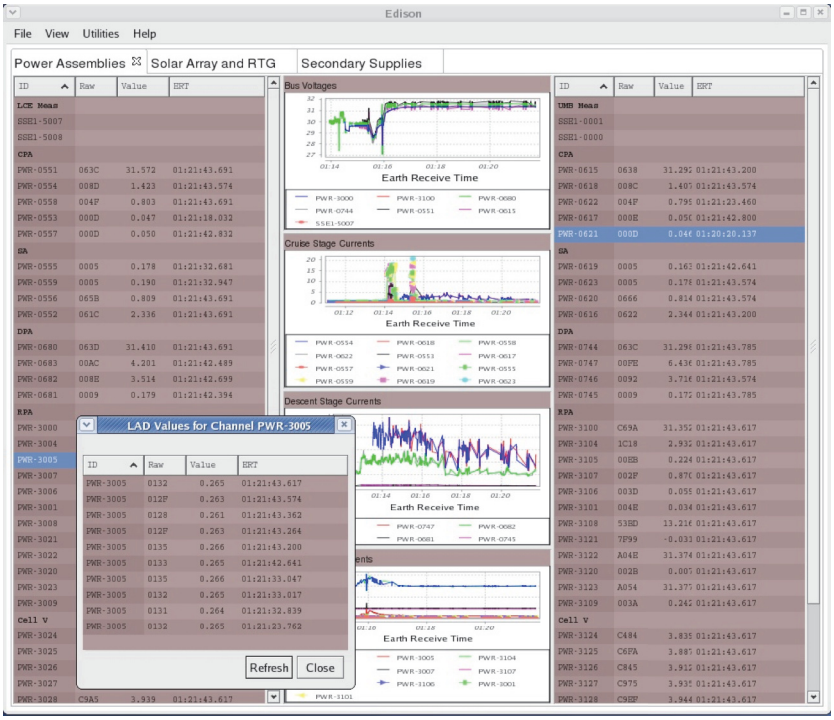

Figure 6. An instance of the AMPCS Monitor application. *Users can customize the GUI to display the telemetry and command data that they are interested in and in their preferred visualization methods. This example also shows the real-time plotting features of AMPCS Monitor.*

Nearly every AMPCS application and tool is designed to be configurable and executed via the host operating system's command-line interface. This allows AMPCS tasks to be scheduled in the operating system level, by using the *cron* job scheduler in Linux for example, to achieve *timed execution* of those tasks. Aforementioned processing of recorded engineering data files is a good candidate for such timed executions. Other jobs that can be scheduled to satisfy the automation requirements, either AMPCS-specific or otherwise, include: fetching of the latest pass list, archiving log data, and various report generation.

E. AMPCS Session Logs

Lights-out automation has its risks. When operation is left unattended and *an anomaly occurs*, rapid identification of both the root problem and the range affected may be more difficult. Troubleshooting and restoring the operation back to normal might be a more complicated effort than if the operator himself had been controlling and observing all activities manually. The SMAP GDS approach to this problem is to ensure that actions performed by automation and events that occur be *logged* informatively, then made available to authorized users both locally and remotely. The operator sitting in front of the console should be able to follow these log messages as they occur in real-time.

AMPCS stores both its application and user-generated logs into the database, and these logs can be queried by session IDs, time ranges, and other filtering criteria. AMPCS time-stamps them just like all telemetry and command objects processed in the system, and this allows telemetry, command, and automation events to be associated and juxtaposed in time-order, aiding analysis and troubleshooting.

Employing as much as possible the capabilities provided by AMPCS and its MTAK API to satisfy automation requirements significantly reduces new development effort. A drawback, however, is that *automation control* becomes fragmented and can become user-unfriendly. As a case in point, the different types of automatable activities shown in Figure 4 may be handled by separate scripts. Those scripts, in turn, may have different execution schedules. The script that handles pass preparation, for instance, is scheduled to execute based on the pass list. On the other hand, recorded engineering data processing and pass list fetch are activities that can be scheduled independently from the spacecraft pass schedule (e.g. check for updated pass list *every 15 minutes*). cron *can* be used to schedule pass-based and other types of jobs. For example, AOE can use separate cron table (*crontab*) files for the different schedule types. To ensure that pass-based jobs are executed according to the latest station schedules, a periodic job can be scheduled to update the *pass-based crontab* using the information from the most recently fetched pass list file. This introduces a form of meta-automation, which makes automation complex and unintuitive to manage. In summary, the AOE design, in its present state at time of writing, has the capability to automate routine real-time operations activities but still lacks a means of centrally managing them in an operator-friendly way. Such centralized automation management would have to provide an interface that allow the operators to easily select and deselect passes from the automation queue, along with other types of scheduled jobs that are part of AOE.

VIII. Case Studies: Jason-1 and WISE

The ultimate goal for command and telemetry automation is to reach "lights-out," reliably performing routine real-time operations unattended by a human operator. Jason-1 and WISE are two missions that have achieved and demonstrated this. We consider their case studies, as they have guided SMAP GDS's automation objectives discussed in previous sections, and because there are further development that remains to bring AOE to maturity before mission operations begin in the year 2014. There are also few but significant differences in the mission characteristics between those two missions and SMAP, and SMAP GDS will have to provide new solutions to accommodate them.

The Jason-1 and WISE projects represent two highly successful JPL missions that have accomplished cost savings through the use of command and telemetry automation. These missions use a command and telemetry system that incorporates a *scripting engine* that provides a software interface to all command and telemetry functions available to a human user. The general approach to automation was to first define operational procedures that were executed manually until the procedures were mature and well understood. These procedures were then used to guide the development of automation scripts. Although complete lights-out operations functionality was developed and demonstrated for Jason-1, the automation is monitored 24/7 by an on-console flight controller. As a result of this approach, the Jason-1 project annually reports data completeness statistics well over *99%*, with 93% of the data provided by automated tracking passes. In general, Jason-1 and WISE were able to achieve significant operations cost savings through automation by maintaining a small operations team of approximately 8 full-time employees.

The command and telemetry automation approach for the SMAP project is being inherited from Jason-1 and WISE. This approach includes the following automated activities that will be performed for each pass: establish station connections before the pass, monitor and alarm EHA telemetry beginning at the start of pass, wait for the elevation mask to be reached before commanding, verify uplink via *No Operation* commands, send routine commands after verifying uplink capability, allow specific manual commanding during or after routine uplinks, disconnect from station at end of pass, verify data completeness using post-pass analysis, prepare data retransmission commands for the next pass if necessary.

Although the automation approach for SMAP is inherited from successful legacy projects, SMAP will encounter specific automation challenges that were not addressed by the previous missions. First, the SMAP project will not receive *station monitor data* from the tracking stations. Station monitor data was extremely beneficial to the Jason-1 and WISE automation by providing valuable insight into ground station performance. The data was also used to determine the health of the pre-pass station connection, and in allowing automation to determine when the *elevation mask* was reached before attempting commanding. The automation for SMAP will have to use alternative strategies to acquire the information provided by station monitor data. In addition to the challenge of not having station monitor data, the SMAP project will be challenged in meeting its data return requirements by not having *commandable downlinks*. For Jason-1 and WISE, data downlink was commanded from the ground automation after the verification of stable command and telemetry links. Data downlink for SMAP will be controlled by an onboard background sequence. This sequence will enable downlink at predetermined times. Any anomaly that affects ground station availability will result in data that will not be acquired at initial downlink. Therefore, the use of data retransmission becomes a more critical aspect of automation for SMAP. Automation for SMAP will have to be robust enough to accommodate a potentially high number of data retransmission requests, including the retransmission of entire tracking passes whose data is missing. However, automation of the retransmission process will allow data recovery that is more timely and cost effective than manual effort. Finally, staffing for SMAP operations will be provided nominally for the standard 8:00 a.m. to 5:00 p.m. work day, five days a week. Automation will be required to perform off-hour and weekend downlink of telemetry data. Relying on automation during these times will result in cost savings in staffing. Although there are significant challenges in providing command and telemetry automation for SMAP, overcoming them will allow the SMAP project to ultimately achieve operations cost savings that are comparable to those achieved by Jason-1 and WISE.

IX. Moving Toward "Lights-Out"

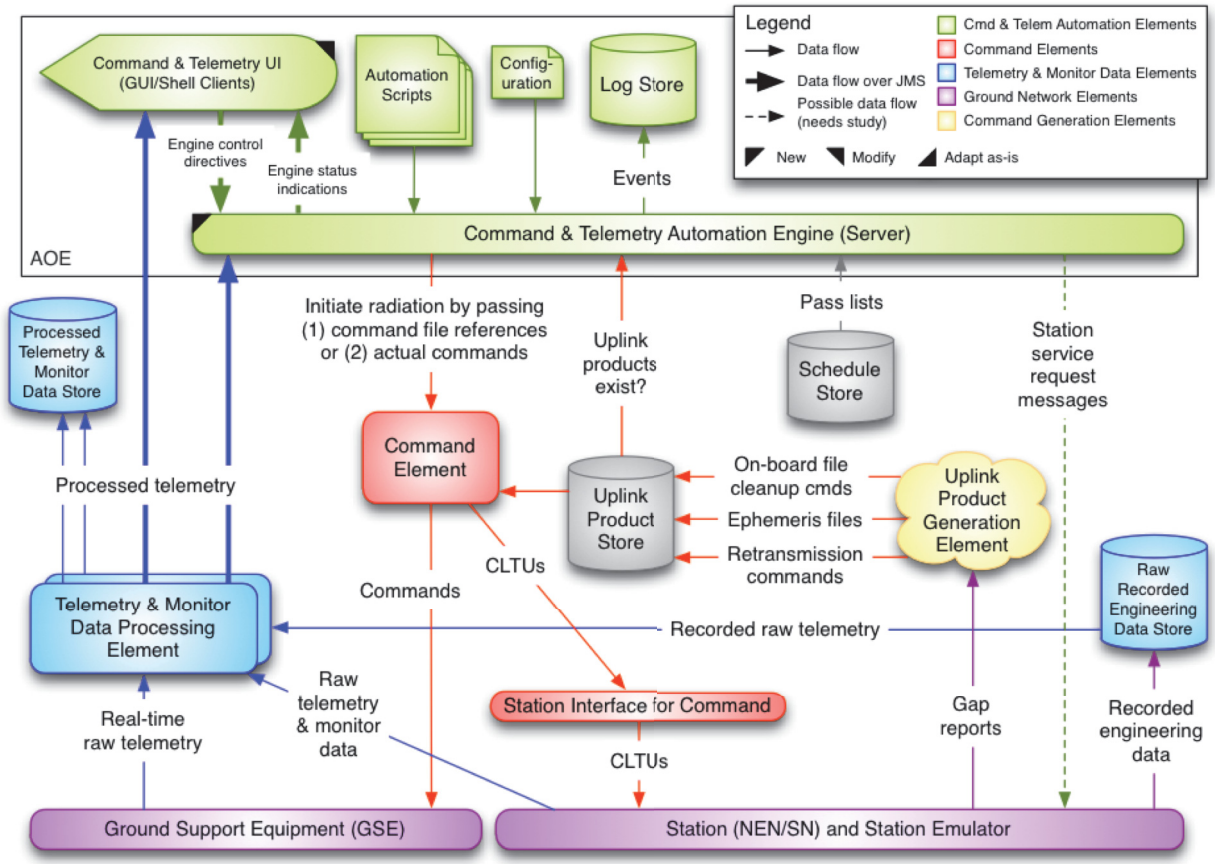

Figure 7. Context diagram showing a possible design for centralized automation control and corresponding user interface. *The* automation engine *would be a new development effort, but the existing AMPCS GUI may be extended to provide a monitoring and control interface specific for automation.*

At the time of SMAP Mission System (MS) Architecture Review in 2010, the baseline automation design included lights-out commanding and science data retransmission capabilities. In 2011, those two functions were descoped from the design to stay within the mission budget. Lights-out commanding is now back into the design and planning for the development effort is under way. This automation feature will allow commanding to be completely scriptable, enabling the automated activity flow discussed earlier in this paper. As regards science data retransmission, it will be put under consideration for possible development should additional funding becomes available. Historical data of station performances show, however, that SMAP's 96% science data completeness requirement can be satisfied even without the retransmission capability.

To address the limitations mentioned previously, such as automation control fragmentation and lack of a user interface for comprehensive management of AOE's automation activities, it is worthwhile to consider how the AOE design may be further developed to solve those issues. Taking a cue from Jason-1 and WISE missions' GDS automation, perhaps an *automation engine* that provides centralized automation management would be beneficial. Figure 7 shows a context diagram of a possible such design.

With the addition of a centralized automation engine and a corresponding user interface, it will be possible to improve the quality of information provided to the users when anomalies occur. That, along with features to isolate problems and contain them (for example, halting a script that received an error code from a shell command, along with all other *scripts that have dependency* on that script) can assist in minimizing downtime and quickly restoring operations back in good order. Without such facilities specifically designed to assist human users, lights-out automation may cause greater operations cost hit when things go wrong.

X. Conclusion

Space missions can save significant costs by automating its routine operations that would normally be performed by human operators. The SMAP mission is committed to keeping both the development and operations costs low, and therefore its GDS is mostly comprised of heritage subsystems that have successfully been used in previous missions. SMAP GDS is using the multimission subsystems AMPCS and CPD to satisfy the mission's telemetry processing and spacecraft commanding needs. Using AMPCS's MTAK API, these functions can be scripted, allowing many real-time operations activities to be automated. Lights-out automation of SMAP's routine telemetry and command operations include: station connection, verification of closed-loop communication with the flight system, updating the ephemeris data on the spacecraft, analyzing telemetry gaps, sending retransmission commands, pass list updates, processing recorded engineering data, and receiving the science data. Using existing capabilities in AMPCS, lights-out telemetry and command automation for the SMAP mission can be achieved. The automation system may be further improved with a centralized automation server for easier scheduling and management of automation tasks. User interface specifically designed for automation can be invaluable when anomalies occur during lights-out automation.

Appendix A
Acronym List

AMMOS	Advanced Multi-Mission Operations System
AMPCS	AMMOS Mission Data Processing and Control System
AOE	Automated Operations Element
API	Application Programming Interface
ATLO	Assembly, Test, and Launch Operations
CLTU	Command Link Transmission Unit
CPD	Command Preparation & Delivery
CPE	Command Processing Element
CSV	Command-Separated Values
DSN	Deep Space Network
EDOS	EOS Data and Operations System
EDPE	Engineering Data Processing Element
EHA	Engineering Housekeeping and Accountability
EOS	Earth Observing System
EVR	Event Record
FSW	Flight Software
GDS	Ground Data System
GSE	Ground Support Equipment
GSFC	Goddard Space Flight Center
GUI	Graphical User Interface
ID	Identification or Identity
JMS	Java Message Service
JPL	Jet Propulsion Laboratory
MOC	Mission Operations Center
MPCS	Mission Data Processing and Control System
MTAK	MPCS Test Automation Kit
NAIF	Navigation and Ancillary Information Facility
NASA	National Aeronautics and Space Administration
NEN	Near Earth Network
SCMF	Spacecraft Command Message File
SDS	Science Data System
SIC	Station Interface for Command
SMAP	Soil Moisture Active Passive
SN	Space Network
SPICE	Spacecraft ephemeris, Planet ephemerides, Instrument description, C-matrix, and Events
SSE	Simulation & Support Equipment
TCP/IP	Transmission Control Protocol/Internet Protocol
WISE	Wide-field Infrared Survey Explorer
WSTS	Workstation Test Set
XML	Extensible Markup Language

Acknowledgments

The authors would like to thank Brian Hammer, Daniel Hurley, Curt Eggemeyer, and Sheldon Shen (all JPL affiliates) for their part in the SMAP GDS pass automation design work described in this paper. Thanks also go out to the past and present members of the AMPCS team for its design and development: Marti DeMore, Jesse Wright, Brent Nash, Ashley Shamilian, Jim McKelvey, Nicole Ameche, William Quach, Michael Tankenson, Kyran Owen-Mankovich, and Lloyd Deforrest.

The work described in this paper was carried out at the Jet Propulsion Laboratory (JPL), California Institute of Technology (Caltech), under a contract with the National Aeronautics and Space Administration (NASA). The work was funded by the Soil Moisture Active Passive (SMAP) project and by the Multimission Ground System and Services Office (MGSS).

References

[1]Committee on Earth Science and Applications from Space: A Community Assessment and Strategy for the Future, National Research Council, *Earth Science and Applications from Space: National Imperatives for the Next Decade and Beyond*, The National Academies Press, Washington, D.C., 2007.

[2]Acton, C.H., "Ancillary Data Services of NASA's Navigation and Ancillary Information Facility," *Planetary and Space Science*, Vol. 44, No. 1, 1996, pp. 65-70.